Karadeniz Teknik Üniversitesi
Bilgisayar Bilimleri Bölümü
BILB2001 Yazılım Geliştirme I

Veri Yapıları I Uygulama Föyü

Hazırlayanlar

Dr. Öğr. Üyesi Tolga BERBER
Arş. Gör. Beyzanur SİYAH

1 Hesap Makinesi

Hesap makineleri temel matematiksel işlemleri yapmak için tasarlanmış elektronik cihazlardır. Bu cihazların çalışma prensibi 0'dan 9'a kadar olan sayıları ve operatörleri bir Yığın (Stack) içerisine eklerek çalışmasından ibarettir. Burada yazılan ifadelerin **postfix** isimli özel bir gösterimi kullanılmaktadır. Aşağıda postfix gösterimleri ile iligli bazı örnekler bulacaksınız.

Normal Gösterim (Infix)	Postfix Gösterimi
2 + 3	2 3 +
2 * 4 + 3	2 4 * 3 +
2 + 3 * 5 + 6	2 3 5 * + 6 +
3 - 4 / 5 + 8 / 6	3 4 5 / - 8 6 / +
(2 + 3) * 5 + 6	2 3 + 5 * 6 +
6 + 7 * 8 - 4 / (2 + 3)	6 7 8 * + 4 2 3 + / -

Tablo 1: Bazı Infix->Postfix dönüşümleri

NOT: Burada *Infix* terimi işlem operatörünün iki sayının ortasında olması durumunu, *Postfix* terimi işlem operatörünün kendinden önceki iki sayı ile ilgilenmesi durumunu göstermektedir.

Algoritma 1 postfix şeklinde girilen bir ifadeyi çalıştırmak amacıyla kullanılabilir.

Bu uygulama çalışmasının ilk bölümünde sizden aşağıdaki görevleri tamamlamanız beklenmektedir.

Görev 1: Algoritma 1'i C++ ile bir fonksiyon olarak tanımlayınız. Fonksiyonun tanımı aşağıdaki gibi olmalıdır.

```
float postfixCalistir(std::string postfixIfade) {

}
```

Görev 2: Bu algoritmayı mod alma (%) ve üs alma (^) operatörlerini destekleyecek hale getiriniz.

```
Veri: p: Postfix İfade
Sonuç: İfade Sonucu
degerler ←Yığın(∅);
for i ← 0 to Uzunluk(p) do
    kr ← p[i];
    if kr ≥ '0' ∧ kr ≤ '9' then
        degerler.ekle(kr);
    else
        sag ← degerler.cikar();
        sol ← degerler.cikar();
        switch kr do
            case '+' do
                degerler.ekle(sag + sol);
            end
            case '-' do
                degerler.ekle(sag - sol);
            end
            case '*' do
                degerler.ekle(sag * sol);
            end
            case '/' do
                degerler.ekle(sag / sol);
            end
        end
    end
end
```

Algoritma 1: Postfix İfade Çalıştırma Algoritması

2 Hastane Acil Servis Kuyruğu Düzenleme

Bir hastanenin acil servisine başvuran hastaların önceliklerinin belirlenmesi için bir yazılım talep edilmektedir. Bu hastaların toplamda 4 farklı öncelik sıralaması söz konusudur. Bu gruplar ile ilgili aşağıdaki bilgiler bilinmektedir.

Öncelik Grubu	Bilinenler
Ö.G. 1	En acil hasta grubudur, günde ortalama 750 hasta bu gruptadır.
Ö.G. 2	Acil hasta grubudur, günde ortalama 800 hasta bu gruptadır.
Ö.G. 3	Normal hasta grubudur, günde ortalama 1500 hasta bu gruptadır.
Ö.G. 4	Ayakta tedavi edilebilen bir hasta grubudur, günde ortalama 2000 hasta bu gruptadır.

Tablo 2: Öncelik Sıralamaları ile ilgili bilinenler

Bu sorunu gidermek için bir C++ sınıfı hazırlayınız. Sınıfınızın prototipi aşağıdaki gibi olmalıdır.

```
class HastaneKuyrugu {
public:
    HastaneKuyrugu() {

    }

    ~HastaneKuyrugu() {

    }

    void hastaGeldi(int OG) {

    }

    int siradakiHasta() {

    }
```

```
    int bekleyenHastaSayisi() {

    }


private:
    int _sonSiraOG1;
    int _sonSiraOG2;
    int _sonSiraOG3;
    int _sonSiraOG4;
    XXXXXX _kuyruktaBekleyenHastalar;
};
```

Bu sınıfta her bir öncelik grubu için ayrı bir sayaç tutulmakta ve bu sayaç o gruba ait bir hasta geldiğinde arttırılıp hastaya bir sıra numarası verilmektedir. **Bu sıra numarasının öncelik numaralarına göre dağıtılması sizin çözmeni gereken bir sorundur.** Örneğin; 32010 sıra numarasının hangi öncelik grubundaki kaçıncı hasta olduğunu sizin bulmanız beklenmektedir. Ayrıca kodda yer alan **XXXXXX** ifadesi yerine uygun veri yapısını kullanmanız beklenmektedir.

Burada ödevlerinizi aşağıdaki şekilde göndermeniz beklenmektedir.

1. Görev için "<Ogr No>_Uygulama2_Gorev1.cpp",
2. Görev için "<Ogr No>_Uygulama2_Gorev2.cpp"

isimli dosyaları oluşturunuz. Bu dosyaları ödev sayfasına ekleyip teslim etmeyi **UNUTMAYINIZ!**